Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет

Рубежный контроль № 2

По курсу «Технологии машинного обучения»

ИСПОЛНИТЕЛЬ:

Горбатенко И.А.

Группа ИУ5-64

_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

_____

"__"_____2020 г.

Москва 2020

Рубежный контроль №1 по курсу "Технологии машинного обучения"

Горбатнко И.А. ИУ5-64

Задание:

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста. Необходимо сформировать признаки на основе CountVectorizer или TfidfVectorizer. В качестве классификаторов необходимо использовать два классификатора, не относящихся к наивным Байесовским методам (например, LogisticRegression, LinearSVC), а также Multinomial Naive Bayes (MNB), Complement Naive Bayes (CNB), Bernoulli Naive Bayes. Для каждого метода необходимо оценить качество классификации с помощью хотя бы одной метрики качества классификации (например, Accuracy). Сделате выводы о том, какой классификатор осуществляет более качественную классификацию на Вашем наборе данных.

Выполнение:

Подключим библиотеки, загрузим набор данных:

In [20]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classi
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_s
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, Lin
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, exp
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRe
from gmdhpy import gmdh
%matplotlib inline
sns.set(style="ticks")
```

In [21]:
```
data_train = fetch_20newsgroups(subset='train', remove=('headers', 'footers
data_test = fetch_20newsgroups(subset='test', remove=('headers', 'footers')
```

In [22]:
```
data_train.target.shape
```

Out[22]:
```
(11314,)
```

In [23]:
```
data_train.data[:3]
```

Out[23]:
```
['I was wondering if anyone out there could enlighten me on this car I sa
w\nthe other day. It was a 2-door sports car, looked to be from the late
60s/\nearly 70s. It was called a Bricklin. The doors were really small. I
n addition,\nthe front bumper was separate from the rest of the body. Thi
s is \nall I know. If anyone can tellme a model name, engine specs, years
\nof production, where this car is made, history, or whatever info you\nh
ave on this funky looking car, please e-mail.',
 "A fair number of brave souls who upgraded their SI clock oscillator hav
e\nshared their experiences for this poll. Please send a brief message de
tailing\nyour experiences with the procedure. Top speed attained, CPU rat
ed speed,\nadd on cards and adapters, heat sinks, hour of usage per day,
floppy disk\nfunctionality with 800 and 1.4 m floppies are especially req
uested.\n\nI will be summarizing in the next two days, so please add to t
he network\nknowledge base if you have done the clock upgrade and haven't
answered this\npoll. Thanks.",
 'well folks, my mac plus finally gave up the ghost this weekend after\ns
tarting life as a 512k way back in 1985.  sooo, i\'m in the market for a
\nnew machine a bit sooner than i intended to be...\n\ni\'m looking into
picking up a powerbook 160 or maybe 180 and have a bunch\nof questions th
at (hopefully) somebody can answer:\n\n* does anybody know any dirt on wh
en the next round of powerbook\nintroductions are expected?  i\'d heard t
he 185c was supposed to make an\nappearence "this summer" but haven\'t he
ard anymore on it - and since i\ndon\'t have access to macleak, i was won
dering if anybody out there had\nmore info...\n\n* has anybody heard rumo
rs about price drops to the powerbook line like the\nones the duo\'s just
went through recently?\n\n* what\'s the impression of the display on the
180?  i could probably swing\na 180 if i got the 80Mb disk rather than th
e 120, but i don\'t really have\na feel for how much "better" the display
is (yea, it looks great in the\nstore, but is that all "wow" or is it rea
lly that good?).  could i solicit\nsome opinions of people who use the 16
0 and 180 day-to-day on if its worth\ntaking the disk size and money hit
to get the active display?  (i realize\nthis is a real subjective questio
n, but i\'ve only played around with the\nmachines in a computer store br
eifly and figured the opinions of somebody\nwho actually uses the machine
daily might prove helpful).\n\n* how well does hellcats perform?  ;)\n\nt
hanks a bunch in advance for any info - if you could email, i\'ll post a
\nsummary (news reading time is at a premium with finals just around the
\ncorner... :( )\n--\nTom Willis  \\  twillis@ecn.purdue.edu    \\    Pur
due Electrical Engineering']
```

In [24]:
```
vectorizer = TfidfVectorizer()
vectorizer.fit(data_train.data + data_test.data)
```

Out[24]:
```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features
=None,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=Non
e,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

In [25]:
```
X_train = vectorizer.transform(data_train.data)
X_test = vectorizer.transform(data_test.data)

y_train = data_train.target
y_test = data_test.target
```

In [26]:
```
X_train
```

Out[26]:
```
<11314x152843 sparse matrix of type '<class 'numpy.float64'>'
        with 1467517 stored elements in Compressed Sparse Row format>
```

In [27]:
```
X_test
```

Out[27]:
```
<7532x152843 sparse matrix of type '<class 'numpy.float64'>'
        with 951914 stored elements in Compressed Sparse Row format>
```

In [28]:
```
def test(model):
    print(model)
    model.fit(X_train, y_train)
    print("accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

In [29]:
```
test(LogisticRegression(solver='lbfgs', multi_class='auto'))
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tr
ue,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose
=0,
                   warm_start=False)
accuracy: 0.774429102496017
```

In [30]:
```
test(LinearSVC())
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
accuracy: 0.8048327137546468
```

In [31]:
```
test(MultinomialNB())
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
accuracy: 0.72623473181094
```

In [32]:
```
test(ComplementNB())
```

```
ComplementNB(alpha=1.0, class_prior=None, fit_prior=True, norm=False)
accuracy: 0.8089484864577802
```

In [33]:
```
test(BernoulliNB())
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
accuracy: 0.5371747211895911
```

Выводы: Метод Complement Naive Bayes задачу многоклассовой классификации в условиях дисбаланса классов решает лучше всего. Также хорошо себя показал метод LinearSVC

In [ ]:
```
```